Method and apparatus for reordering mem ry operations in a superscalar or very l ng instruction word processor

Patent Number: EP0742512, A3

Publication date: 1996-11-13

Inventor(s): MORENO JAIME HUMBERTO (US); EBICIOGLU MAHMUT KEMAL (US); LUICK DAVID ARNOLD (US); SILBERMAN GABRIEL MAURICIO (US); WINTERFIELD PHILIP BRAUN (US)

Applicant(s):: IBM (US)

Requested Patent: JP8314721

Application Number: EP19960106145 19960419

Priority Number (s): US19950435411 19950510

IPC Classification: G06F9/38

EC Classification: G06F9/38E2, G06F9/38D4, G06F9/38H2

Equivalents: JP3096423B2, US5625835

Abstract

A method and apparatus for reordering memory operations in superscalar or very long instruction word (VLIW) processors is described, incorporating a mechanism that allows for arbitrary distance between reading from memory and using data loaded out-of-order, and that allows for moving load operations earlier in the execution stream. This mechanism tolerates ambiguous memory references. The mechanism executes only one additional instruction for disambiguation purposes, thus producing good performance, and integrates memory disambiguation with speculative execution of instructions. The overhead introduced is only one instruction, and the load operation can be arbitrarily moved earlier in the instruction stream. The mechanism can cope with conflicts that occur as a result of an unexpected combination of store/load instructions, can be used in a coherent multiprocessor context, and combines speculative execution with reordering of memory operations in a way which requires simple hardware support.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平8－314721

(43)公開日　平成８年(1996)11月29日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/38	３５０		Ｇ０６Ｆ　9/38	３５０Ｘ	

審査請求　未請求　請求項の数３　ＯＬ　（全 12 頁）

(21)出願番号　特願平8－98035

(22)出願日　平成８年(1996)４月19日

(31)優先権主張番号　４３５４１１

(32)優先日　1995年５月10日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　マフムット・ケマール・エビシオグルー
アメリカ合衆国10589ニューヨーク州　ソマーズ　クリスタル・ドライブ　38

(74)代理人　弁理士　合田　潔　（外２名）

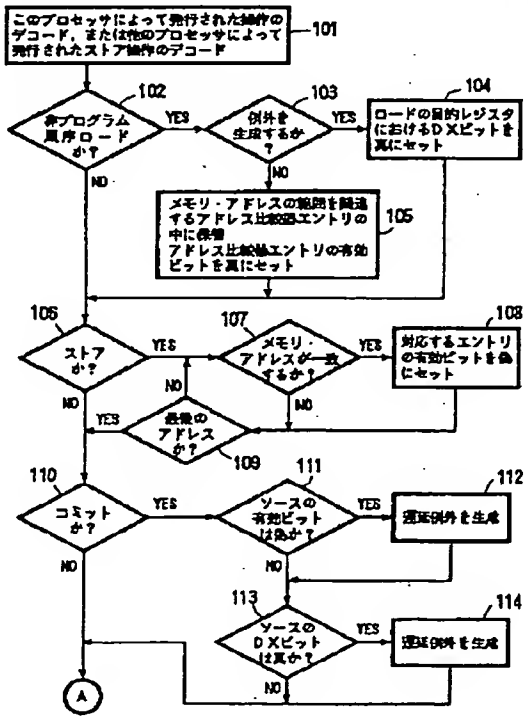
最終頁に続く

(54)【発明の名称】スーパースカラまたはＶＬＩＷプロセッサにおけるメモリ操作の順序替えのための方法及び装置

(57)【要約】

【課題】スーパースカラまたはＶＬＩＷプロセッサの中のメモリ操作を順序替えするための方法および装置を提供する。

【解決手段】本発明による方法および装置は、メモリからの読み出しと非プログラム順序でロードされたデータの使用の間に任意の距離を置くことを可能にしまたロード操作の実行ストリームの前のほうへ移動することを可能にする機構を含む。この機構は、曖昧さをなくすために一つだけの追加の命令を実行するため、すぐれた性能をもたらし、メモリの曖昧さの解消と命令の投機的実行を統合する。導入されるオーバーヘッドは、命令一つだけであり、ロード操作は、命令ストリームの中の前のほうへ任意に移動させることができる。この機構は、ストア／ロード命令の予期しない組合わせの結果として生じる衝突に対処することができ、コヒーレントな多重プロセッサ・コンテキストの中で使用することができ、簡単なハードウエア・サポートを必要とする方法で投機的実行をメモリ操作の順序替えと組み合わせる。



1

【特許請求の範囲】

【請求項１】スーパースカラまたはＶＬＩＷプロセッサにおいてメモリ操作を順序替えする方法であって、

プロセッサによって発行された命令をデコードするステップと、

デコードされた命令が非プログラム順序ロード命令であるかどうかを判別し、そうであれば、その非プログラム順序ロード命令が例外を生成するかどうかを判別するステップと、

例外を生成する非プログラム順序ロード命令について、前記ロード命令の目的レジスタに関連する遅延例外ビットをセットするステップと、

例外を生成しない非プログラム順序ロード命令のメモリ・アドレスをアドレス比較器に保管し、前記アドレス比較器に保管されたメモリ・アドレスのための有効ビットをセットするステップと、

デコードされた命令がストア操作であるかどうかを判別するステップと、

デコードされたストア命令によって参照されたメモリ・アドレスの範囲を前記アドレス比較器の中のすべてのエントリと比較するステップと、

前記アドレス比較器の中の一致する各エントリについて、その対応するエントリの有効ビットを無効にセットするステップと、

デコードされた命令がコミット操作であるかどうかを判別するステップと、

前記デコードされたコミット操作の目的レジスタに関連するアドレス比較器エントリの前記有効ビットをチェックし、前記有効ビットが無効にセットされている場合には遅延例外を生成し、同時に、前記コミット操作のソース・レジスタの遅延例外ビットをチェックし、遅延例外ビットがセットされている場合には、遅延例外を生成するステップと、

例外命令を打ち切り、制御を例外ハンドラへ移すステップと、を有する方法。

【請求項２】許請項１に記載のスーパースカラまたはＶＬＩＷプロセッサにおいてメモリ操作を順序替えする方法であって、さらに、

前記デコードされた命令が非プログラム順序ロード、ストア、又はコミット命令以外であるかどうかを判別するステップと、

該命令によって用いられたすべてのソース・レジスタのための前記遅延例外ビットをチェックし、任意の遅延例外ビットがセットされている場合には、目的レジスターの対応する遅延例外ビットをセットするステップと、を有する方法。

【請求項３】メモリ操作を順序替えすることのできるスーパースカラまたはＶＬＩＷプロセッサであって、

前記プロセッサによって発行される命令をデコードするためのデコーダと、

2

各々が特殊レジスタとしてアクセス可能な遅延例外ビットを有する複数のレジスタと、

デコードされた命令が非プログラム順序ロード命令であるかどうかを判別し、そうであれば、該非プログラム順序ロード命令が例外を生成するかどうかを判別する機能的手段であって、例外を生成する非プログラム順序ロード命令について前記ロード命令の目的レジスタに関連する遅延例外ビットをセットする機能的手段と、

例外を生成しない非プログラム順序ロード命令のメモリ・アドレスを保管するためのアドレス比較器であって、保管されたメモリ・アドレスのためにセットされる有効ビットを有するアドレス比較器と、ただし、

前記機能的手段は、デコードされた命令がストア操作であるかどうかを判別し、

前記アドレス比較器は、デコードされたストア命令によって参照されたメモリ・アドレスの範囲を前記アドレス比較器の中のすべてのエントリと比較し、前記アドレス比較器の中の一致する各エントリについて、その対応するエントリの有効ビットを無効にセットし、

前記機能的手段は、デコードされた命令がコミット操作であるかどうかを判別し、該デコードされたコミット操作のソース・レジスタに関連するアドレス比較器エントリの前記有効ビットをチェックし前記有効ビットが無効にセットされている場合には遅延例外を生成し、また同時に、前記コミット操作のソースレジスタの遅延例外ビットをチェックし、遅延例外ビットがセットされている場合には、遅延例外を生成して例外命令を打ち切り、

例外命令が打ち切られるときに回復コードを実行する例外ハンドラーし、を有するスーパースカラまたはＶＬＩＷプロセッサ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、一般的には、プログラムにおける命令レベルの並列化を利用するために、スーパースカラあるいはＶＬＩＷ（ｖｅｒｙ　ｌｏｎｇ　ｉｎｓｔｒｕｃｔｉｏｎ　ｗｏｒｄ）プロセッサにおけるメモリ操作の順序替えに関し、より詳しくは、任意に分離されたあるいは曖昧なメモリ参照にもかかわらずメモリ操作を順序替えし、それによってコンピュータ・システムの性能を著しくな改善するための方法および装置に関する。該方法および装置は、単一プロセッサ・システムおよび多重プロセッサ・システムに適用が可能である。

【０００２】

【従来の技術】高性能の現代のプロセッサは、プログラムにおける命令レベルの並列化を利用するために、すなわち同時に二つ以上の命令を実行するために、スーパースカラおよび／またはＶＬＩＷ技術に依存している。このようなプロセッサは、多数の機能ユニットを含んでおり、命令の順次ストリームを実行し、一サイクルごとに

メモリから二以上の命令を取り出すことができ、また、リソースの依存性および利用可能性に応じてサイクルごとに二以上の命令をディスパッチすることができる。これらの機能は、コンパイラによって活用され、コンパイラは、スーパースカラおよび／またはＶＬＩＷ機構のために最適化されたコードを生成する。

【０００３】順次プログラムにあっては、メモリ・ロード操作は、メモリからデータを読み出し、それをプロセッサ・レジスターにロードし、ロードされたデータに依存する一連の操作を頻繁に開始する。利用可能なリソースがあるスーパースカラあるいはＶＬＩＷプロセッサでは、メモリ・ロード操作をできるだけ早期に開始するとが有利である。なざなら、他の遊休のリソースを使用できるようになり、（潜在的なキャッシュ・ミスを含めて）メモリ・アクセスの遅延を隠し、それによってプログラムの実行時間が短縮できるからである。ロードならびにそのロードに依存する操作は、厳密な順次プログラムで行なわれるより早期に実行され、実行時間の短縮が達成される。このためには、非ブロック化ロード（すなわち、キャッシュ・ミスを生みだすロードを越えて命令を発し続ける）を実行する機能、先行するストアの前にロード（すなわち、非プログラム順序のロード）を発行する機能、先行するブランチの前へロードを移動させる（すなわち、投機の）機能、およびあるロードに依存する操作を他の操作の前へ移動させる機能が必要である。言い換えれば、プログラムの操作の順序替えを行なう機能が必要である。

【０００４】メモリ操作の順序替えを行なう機能は、いくつかの要因、とくにプログラムの実行における実行時依存性から生じる要因によって制限される。これらの要因には、条件付きブランチ命令の前へ操作を移動することおよび曖昧なメモリ参照が含まれる。

【０００５】ある操作を先行する条件付きブランチ命令の前へ移動することは、その操作が本当に必要かどうかがわかる前に実行されるため、プログラムの実行に投機性を招くことになる。その操作が必要であろうという期待のもとで、コードの移動が行なわれる。副作用のないレジスタ間操作は、その結果が未使用（「死んだ」）レジスタに保管される限り、投機的に実行することができる。操作の必要がない場合には、その結果は単に無視される。他方、副作用のあるレジスタ間操作及びメモリ・ロード操作は、例外（エラー）、保護違反、あるいは揮発性メモリ位置へのアクセスなど、起きてはならない副作用から回復する機構が存在している場合にのみ投機的に実行することができる。

【０００６】あるメモリ・ロード操作を先行するメモリ・ストア操作の前へ動かすことは、コンパイル時にロードとストアによってアクセスされるメモリ位置が異なることを判別できない場合には、プログラムの実行において、曖昧な参照という問題が生じる。曖昧でないメモリ

参照は、衝突しないので、非プログラム順序で実行することができる。他方、曖昧なメモリ参照は、衝突の可能性を検出し、定刻前にロードされたデータを無視し、ストア操作が行なわれた後に正しい値を再ロードする機構が存在する場合にのみ非プログラム順序で実行することができる。この衝突は、複数バイトのオペランドの一バイトで起こる場合があるため、ロード操作を行なうことが可能になる前にストア操作を完了することが必要になる。

【０００７】上に述べた二つの問題は異なるものであるが、その影響と要求は同じである。すなわち、曖昧さの副作用を検出しまたそこから回復するための機構が存在しなければならない。以下の説明では、これらの問題は、いずれも「順序替えメモリ・アクセスの問題」と呼ぶ。

【０００８】現在用いられているコンパイルの手法は、メモリ操作の順序替えのための静的メモリの曖昧さをなくすアルゴリズムを含むものである。これらのアルゴリズムは、二つのメモリ参照、すなわち一つのメモリ・ストア操作に続く一つのメモリ・ロード操作が同じ位置にアクセスするかどうかを判別する。参照が衝突しなければ（すなわち、それらが異なるメモリ位置をアドレス指定する場合には）、操作を順序替えして、ロードをストアの前に実行することが可能である。静的な曖昧さをなくす作業は、メモリ・アクセスのパターンが予測可能な場合にのみうまく機能する。しかし、そうでない場合も多く、コンパイラ／プログラマは、参照が実際に衝突することについて控えめな仮定を立て、順次（最初の順序で）実行してプログラムにおける命令レベルの並列化の可能性を低くするようにしなければならない。

【０００９】メモリ操作の順序替えには、これまでにも強い関心がはらわれてきた。例えば、Ｋ．ディフェンドーフおよびM．アレンの論文「モトローラ８８１１０スーパースカラＲＩＳＣマイクロプロセッサの機構」ＩＥＥＥ　Ｍｉｃｒｏ、１９９２年４月、ｐｐ．４０－６３を参照されたい。モトローラ８８１１０プロセッサ中の動的スケジューラは、ストア命令をストア待ち行列にディスパッチしストアされるべきオペランドが他の操作でまだ生成されていない場合にはそのストア操作は停止する。その後のロード命令は、そのストアを迂回してただちにメモリにアクセスし、メモリ・アクセスの動的順序替えを行なうことができる。アドレス比較器は、アドレスの危険を検出し、ロードがストアの前に同じアドレスへ進むのを防ぐ。待ち行列は、三つの未解決のストア操作を保持するため、この構造によって、きついループの実行時オーバーラップが可能となる。この構造は、順次実行のストリームの中でロードを実際に前へ動かすものではなく、ストア操作がが停止されたことによってロード操作が遅延することがないようにするだけである。

【００１０】一定の条件下でのロード／ストア操作のル

5

ープから外への静的な動きは、K．エプシオグルー、R・グロープス、K．キム、G．シルバーマン、およびI．ジブの「スーパースカラ環境内でのＶＬＩＷコンパイレーションの手法」プログラミング言語の設計と実施に関するＳＩＧＰＬＡＮ会議（ＰＬＤＩ’９４）、１９９４年、に記載されている。このアプローチは、基本的には、条件付きで実行されるロードとストアが安全とみなされる場合にはそれらを移動させる追加の機能を有する、ループ不変命令のループから外への静的な移動の一般化である。この最適化の作業のために必要な条件は、衝突するメモリ参照（曖昧なメモリ参照）の可能性がないことを保証することを含むが、これは、常に可能なことではない。

【００１１】プロセッサのアーキテクチャを修正することなしに投機的ロードのスケジューリングを可能にするコンパイレーションの手法は、D．バーンステン、M．ロデー、およびM．ホプキンズの米国特許出願「コンピュータのための命令スケジューラー」（１９９２年５月１４日出願、第０７／８８２７３９号の継続出願としての、１９９４年１２月２７日出願、第０８／３６４８３６号、本出願の譲受人に譲渡）に記載されている。このアプローチでは、投機的実行のためのロード操作の適当性は、それを、その操作で使用されるベース・レジスタに適用される条件および／またはそのベース・レジスタの内容に応じた一定数のカテゴリに分類することによって判別される。すなわち、上記のK．エプシオグルー等が記載している手法と同様に、このアプローチも、コンパイル時に検出できる場合に限定される。

【００１２】「投機的な曖昧さの除去」とよばれるハイブリッドなメモリの曖昧さをなくす手法が、A．フアン、G．スレーブンバーグ、およびJ．シェンの「投機的な曖昧さの除去：ダイナミック・メモリの曖昧さをなくすためのコンパイレーションの手法」第２１回コンピュータ・アーキテクチャ国際シンポジウム、シカゴ、pp．２００－２１０、１９９４年、に提案されている。このアプローチは、ハードウエアとコンパイラの手法を組み合わせてその目的を達成しようとするものである。この手法は、ハードウエア内のガードされた実行機能を必要とする曖昧なメモリ参照のいずれかの結果を見越してコードに変形を行なうものである。各対の曖昧なメモリ参照ごとに、コンパイラは、メモリ参照に依存するコードの二つのバージョンを生成する。一つのバージョンは、アドレスが重なり合うことを仮定し、他のバージョンは重なり合わないことを仮定する。いずれのバージョンにおいても、副作用のない操作が行なわれ、副作用のある操作は、二つのアドレスの比較の結果によってガードされる。このアプローチは、ガードされた実行の機能に加えてもとのプログラムより多くの操作とリソースを必要とし、曖昧さの除去しか行なわず、ブランチの前へロード操作を移動させる機能をもたない。

6

【００１３】ロード操作をストア操作の前に実行できるようにしてプログラム実行のコンパイラの最適化を行なう他の方法が、A．ニコローの「実行時の曖昧さの除去：静的に予想できない依存性への対処」ＩＥＥＥ議事録第３８巻、１９８９年５月、に記載されている。このアプローチは、ストア操作の前へ移動させることのできるロードのコンパイラー識別と必要なコードのコンパイラ挿入に依存するもので、上記資料にA．フアン等が記載しているように、プロセッサがロードとストア操作の諸アドレスの間に一致があるかどうか実行時にチェックを行なうことができるが、保護された実行機能はない。一致がなければ、プロセッサは、ロードがストアの前へ動かされた一連の命令を実行する。他方、一致があれば、プロセッサは、ロードがストアの後に行なわれる一連の命令を実行する。アドレスの一致のチェックがプロセッサによって行なわれるので、このアプローチは、より多くの命令の実行とそれに関連する依存性（例：メモリ・アドレスの明示生成とアドレス比較）による潜在的な性能劣化を招く。さらに、順序替えロード操作は、ロードおよびストアの両操作のためのメモリ・アドレスが解決されるまで行なうことができない。

【００１４】非プログラム順序操作の性能を改善するための方法および装置は、M．クマール、K．エプシオグルー、およびE．クロンスタッドの米国特許出願「コンピュータ・システムの性能を改善するための方法および装置」（１９９２年５月６日出願第０７／８８０１０２号の継続出願としての、１９９４年１０月７日出願、第０８／３２０１１１号、本出願の譲受人に譲渡）に記載されている。この方法およびアプローチは、コンパイラの手法、４つの新しい命令、およびアドレス比較装置を使用するものである。コンパイラは、メモリ・ロード操作をメモリ・ストア操作の前へ静的に移動させる。非プログラム順序でロードされたオペランドのアドレスは、アソシアティブ・メモリへ保管される。要求があれば、アドレス比較装置が、アソシアティブ・メモリに保管されたアドレスをストア操作によって生成されたアドレスと比較する。衝突が検出されれば、問題を解決するための回復コードが実行される。このシステムは、それらのアドレスの比較を行なう必要がなくなれば、アソシアティブアドレス内に保管されたアドレスを消去する。このアプローチは、メモリ操作の順序替えの問題を扱うだけのものである。メモリ・ロード操作を投機的に実行する機能は含んでいない。さらに、このアプローチは、アドレス内の衝突のチェックを始動させまた必要のなくなったオペランドのアドレスを消去するための特別の命令を必要とし、また、コンパイラに潜在的な衝突をすべて検出してペアリングさせなければならない。その結果、このアプローチは、（おそらくはエラーによって生じる）ストア／ロード命令の予想されない組合わせによって生じる衝突に対処することができないし、また、コヒーレ

ントな多重プロセッサのコンテキストで使用することもできない。

【0015】関連する主題として、コンパイラ・サポートをともなったハードウエア機構が、K．エブシオグルー、およびM．ジルバーマンの米国特許出願「投機的命令における例外の取り扱い」（1995年1月24日出願、第08／377563号、本出願の譲受人に譲渡）に記載されている。この機構は、投機的に実行される命令によって生じる例外からオーバーヘッドを少なくするものである。この機構は、投機的な命令の実行中に生成された例外を示すためのレジスタごとの追加ビット、レジスタ・オペランドを保管して例外によって無効化された投機的命令が再実行されるようにするための二つの付加的レジスタ・ファイル、ならびに例外の発生源をたどることのできる情報などのハードウエアに依存するものである。この機構は、投機的命令にのみ適用可能で、順序替えされたメモリ操作には適用できない。

【0016】F．アマーソン、R．ガプタ、V．カタール、M．シュランスカーの特許出願「ロード命令の積極的実行を可能にするメモリ・プロセッサ」（1993年4月2日出願、英国特許出願、GB2265481A、第9302148．3号）には、ロード命令を順序替えするための装置および方法が記載されている。この特許出願は、コンパイラが、待ち時間の長いロード命令を命令の順序の前の方へ移動させるコンピュータ・システムのためのメモリ・プロセッサを記載したものである。このメモリ・プロセッサは、ロード命令を、ロードに先立って実行されたであろうなんらかの順序が後のストア命令がそのロード命令によって指定されたと同じアドレスを参照するかどうかを判別するための十分な時間だけ特別のレジスタ・ファイルに保管する。その場合には、メモリ・プロセッサが、最初のロードを命令ストリームの中に再挿入し、それが順序通りに実行されるようにする。したがって、このシステムでは、コンパイラの制御のもとでロードをストアの前へ移動させることができるもので、ハードウエアに依存して衝突から回復するためのコードを挿入する。しかし、このシステムでは、ロードに依存する他の命令の順序替えを行なうことはできない（ハードウエア・リソースは、ロード命令のみを再挿入することができる）。また、ロードまたは他の命令の投機的実行を行なうこともできない。言い換えれば、この方法および装置は、コンパイルするときに最大値を知る必要のあるロード命令の待ち時間を隠すことに限られている。

【0017】

【発明が解決しようとする課題】したがって、本発明の一つの目的は、ロード命令を実行ストリームの前の方へ移動することができ、メモリからの読み出しと非プログラム順序でロードされたデータを用いることの間に任意の距離を置くことができる機構を提供することである。

本発明の他の一つの目的は、ロード操作のループから外への移動に限定されずまた曖昧なメモリ参照を許容することのできる機構を提供することである。本発明のさらに他の一つの目的は、曖昧さをなくすために一つだけ追加の命令を実行し、それにより性能を高め、またメモリの曖昧さの解消を投機的実行と統合する機構を提供することである。本発明のさらに他の一つの目的は、導入されるオーバーヘッドが命令一つだけであり、ロード操作を任意に命令ストリームの前のほうへ移動させることのできる機構を提供することである。本発明のさらに他の一つの目的は、ストア／ロード命令の予想しない組合わせの結果生じる衝突に対処することができ、コヒーレントな多重プロセッサのコンテキストで使用することのできる機構を提供することである。本発明のさらに他の一つの目的は、必要な実装がかなり簡単な、投機的実行とメモリ操作の順序替えを組み合わせる機構を提供することである。

【0018】

【課題を解決するための手段】本発明にもとづけば、任意に分離された曖昧なメモリ参照についてでも、スーパースカラまたはVLIWプロセッサにおけるメモリ操作の順序替えを行なうための方法および装置が提供される。この順序替えは、従属操作の順序をプログラム実行の前のほうへ移動させることによってプログラムの経路長を短縮し、それによってコンピュータ・システムの性能を高めるものである。この方法および装置は、メモリ操作の順序替えと投機的実行を統合するものであり、単一プロセッサ・システムにもまた多重プロセッサ・システムにも適用が可能である。装置は、メモリのアドレス指定における衝突をチェックする多重入力アドレス比較器、順序替えされたメモリ操作によって生成された衝突を示す比較器入力ごとの状況ビット、保留の例外を示すためのプロセッサのレジスタ・ファイル内のレジスタごとの状況ビット、レジスタを非プログラム順序でロードし、非プログラム順序でロードされたレジスタをコピーし、非プログラム順序でロードされたレジスタをコミットするための特別な命令、およびこれらのリソースを使用するコードならびに非プログラム順序で命令を実行する間に生じる例外から回復するためのコードを生成するためのコンパイラ・サポートから構成される。

【0019】非プログラム順序のメモリ操作から、下記の要件が生じる。

・非プログラム順序ロード操作によって生成される例外（副作用）は、ロードされたデータがプログラム順序での（非投機的）操作に使用されるまで報告されてはならない（行なわれてはならない）。

・ロード操作をストア操作の上へ移動させたときのアドレスのオーバーラップによる衝突が検出されなければならない。

・ストアの前にロードされたデータは、プログラム順序

で使用される前に妥当性がチェックされなければならない（言い換えれば、オーバーラップするストアにより無効となっていないことのチェック）。

・揮発性の位置には投機的にロードすることができない。

【００２０】本発明で用いられるアプローチは、下記に依存するものである。

・命令レベルでの並列化を利用するためのコンパイラによるコードの静的順序替え。

・曖昧なメモリ・アクセスにおける衝突を検出し、遅延した例外を報告し、非プログラム順序でロードされたデータを操作するためのハードウエア・サポート。

・非プログラム順序でロードされたデータを操作しまた遅延した例外から回復するためのコードのコンパイラ生成。

【００２１】

【発明の実施の形態】図１および図２は、本発明によって行なわれる非プログラム順序および他の操作のフローチャートを示す。

【００２２】プロセッサが発行する各命令およびコヒーレントな多重プロセッサ・システムにおける他のプロセッサが発行する各ストア操作は、機能ブロック１０１でデコードされる。命令が、決定ブロック１０２で非プログラム順序ロード操作と判別され、該命令が、決定ブロック１０３で例外を生成すると判別された場合、機能ブロック１０４で、ロード命令の目的レジスタに関連する遅延例外（ＤＸ）ビットがセットされるが、プロセッサにはいかなる例外も発生されない。他方、該命令が例外を生成しない場合には、該命令によって参照されるメモリ・アドレスの範囲がアドレス比較器（ＡＣ）のエントリに保管され、このエントリの有効なビットが、機能ブロック１０５で有効にセットされる（すなわち、アドレス比較器のエントリは、最近非プログラム順序でロードされたメモリ・アドレスのキャッシュとして作用する）。

【００２３】命令が、決定ブロック１０６でストア操作と判別された場合、決定ブロック１０７で、該命令によって参照されるメモリ・アドレスの範囲がアドレス比較器の中のすべてのエントリと比較される。機能ブロック１０８では、この範囲と一致する各エントリについて、対応する有効ビットを無効にセットする。決定ブロック１０９では、すべてのアドレスが比較されたかどうか判別する。

【００２４】該命令が、決定ブロック１１０でコミット操作と判別された場合、命令のソース・レジスタに関連するアドレス比較器エントリ内の有効ビットが決定ブロック１１１でチェックされる。該ビットが偽にセットされている場合、機能ブロック１１２で遅延例外が生成される。同時に、そのコミット命令のソース・レジスターの遅延例外ビットも決定ブロック１１３でチェックされる。このビットがセットされた場合、機能ブロック１１４で遅延例外が生成される。

【００２５】決定ブロック１１５で、操作が他のいずれかの操作であると判別された場合、決定ブロック１１６で該命令のすべてのソース・レジスタの遅延例外ビットがチェックされる。これらのビットのいずれかがセットされた場合、機能ブロック１１７で、命令の目的レジスタの遅延例外ビットがセットされるが、プロセッサにはいかなる例外も発生されない。それ以外では、機能ブロック１１８で、命令の目的レジスターの遅延例外ビットが偽とセットされる。

【００２６】決定ブロック１１９で遅延例外が判別されてプロセッサに発生された場合、該例外命令は打ち切られ、実行制御が機能ブロック１２０で例外ハンドラへ移される。この例外ハンドラは、該例外を生成しまた該例外が発生される前に実行されたロード操作ならびにロードに依存するすべての操作の実行を繰り返す「回復コード」の実行を担当するものである。

【００２７】図１および図２のフローチャートには、本発明の以下の特徴が示されている。

例外報告：　ロード操作の非プログラム順序実行の間に生じるエラー（副作用）（保護違反など）は、非プログラム順序でロードされたデータが順次命令ストリームの中のロード命令の元の場所でプログラム順序の操作に必要とされるまでは報告されない。エラーがある場合には、ロード命令ならびに非プログラム順序で実行されたロードに依存するすでに実行された他の命令がその地点で再実行される。

【００２８】例外を生じるおそれのある非プログラム順序ロード命令を実行するため、目的レジスタには「遅延例外」ビットのタグが付けられる。このビットは、遅延の形で、非プログラム順序ロード命令の実行中に発生した例外を報告するために用いられる。遅延例外ビットは、非プログラム順序ロードが例外を生成したときにセットされ、レジスタが他の操作に使用されるときに伝播される。コミット操作は、そのオペランドの遅延例外ビットをチェックする。遅延例外ビットがセットされる場合には、遅延例外が生成される。例外ハンドラは、遅延の形で報告される例外を生じたロード命令ならびにそれに依存するすでに実行された他の命令の再実行を担当するものである。

【００２９】記憶・アドレスのオーバーラップによる衝突：　ストア操作の前へ動かされたロード操作は、メモリ・アドレスのオーバーラップによる衝突を生じることがある。衝突は、また、コヒーレントな多重プロセッサ環境の中で他のプロセッサによって行なわれる操作によって生じることもある。いずれの場合にも、非プログラム順序ロードは、非プログラム順序ロードの実行からロードされたデータの最初のプログラム順序の使用までの間に（同じまたは別のプロセッサによる）同じメモリ位

置へのストア操作の結果として無効になったデータにアクセスすることがある。

【００３０】ロード／ストア・アドレスのオーバーラップから生じる衝突は、多重エントリ「アドレス比較器」（ＡＣ）を用いて動的に検出される。非プログラム順序ロード操作を実行するときには、ロードされたオペランドの実アドレスの範囲がアドレス比較器エントリ内に保管される。

【００３１】ストア操作が実行されるときには、そのストア・オペランドの実アドレスの範囲がアドレス比較器内のすべてのエントリの内容と比較される。同様に、コヒーレント多重プロセッサ・システムでは、システム内の他のプロセッサからストア参照を受け取ったときにアドレス比較器エントリもチェックされる。各アドレス比較器エントリについて、その実アドレスにおけるオーバーラップが検出された場合には、そのエントリは無効のマークが付けられる。

【００３２】非プログラム順序でロードされたデータのコミット：　非プログラム順序でロードされたオペランドならびにそれから生じるすべての値は、非投機的（プログラム順序）命令の中のオペランドとして使用できるようになる前に「コミットされる」必要がある。すなわち、プログラムは、特定のレジスターに関連するアドレス比較器エントリがデータが使用される地点（通常は、プログラム内の元の場所）でまだ有効であることを検査する必要がある。このためには特別の命令が用いられ、非プログラム順序でロードされたデータを別のレジスタにオプションでコピーし、同時に関連するアドレス比較器エントリの有効性を検査する。エントリが有効である場合には、コミット（コピー）操作が進行する。一方、エントリが無効である場合には、遅延例外が発生される。例外ハンドラは、そのロード操作および該ロードに依存するすでに実行された他のすべての操作を再実行する。

【００３３】揮発性ロード：　揮発性の位置からのロードは、非プログラム順序で実行されない。（記憶保護機構によって検出された）揮発性の位置からの非プログラム順序にロードするいかなる試みも、単に関連するレジスタの遅延例外ビットを設定するだけである。記憶装置へのアクセスは行なわれない。

【００３４】ハードウエアの実装

図３は、上に述べた非プログラム順序ロードおよび他の操作をサポートするハードウエア・リソースが配備されたプロセッサを示す。このプロセッサは、固定および浮動小数点算術論理装置（ＡＬＵ）など複数の機能ユニットを含む。図には、６個の機能ユニット２０１ないし２０６が示されているが、当業者には、具体的なプロセッサの設計に応じてより多くのまたはより少ない機能ユニットを配備できることが理解されよう。これらの機能ユニットは、データ・キャッシュ２０７からデータにアクセスし、多重プロセッサ・システムの場合には、該データ・キャッシュが他のプロセッサに接続されている。機能ユニットは、汎用レジスタ（ＧＰＲ）２０８、浮動小数点レジスタ（ＦＰＲ）２０９、および条件レジスター２１０に適当に接続される。

【００３５】これまでに記載した構造は、従来のもので当業者にはよく理解されているものである。本発明は、ＧＰＲ２０８、ＦＰＲ２０８、および条件レジスタ２１０にそれぞれアドレス比較器（ＡＣ）バッファ２１１および遅延例外（ＤＸ）ビット２１２、２１３、２１４をそれぞれ付加するものである。より詳細には、一つの遅延例外ビットが非プログラム順序で実行された操作の目的となりえる各レジスタに関連付けられる。遅延例外ビットは、特殊レジスタとしてアクセス可能であり、また、コンテキストの切り替え時にプロセッサの状態の一部として保管され復元される。

【００３６】アドレス比較器エントリは、非プログラム順序でロードされる各レジスタと関連付けられる。図３は、静的関連を示しており、各レジスタは、固有の（固定された）関連エントリをもっている。（あるいは、アドレス比較器エントリは、実行時に、それを必要とする各レジスタに動的に割り当てることもできる。）本発明のこの実施形態にあっては、各ＡＣエントリは、（１）非プログラム順序でロードされたオペランドの実アドレスの範囲、（２）非プログラム順序でロードされたオペランドのいずれかのバイトが同じプロセッサまたはコヒーレントな多重プロセッサ・システムの他のプロセッサのいずれかからのその後のストア操作によって修正されたかどうかを示す有効ビット、および、（３）ＡＣエントリでカバーされるアドレスの範囲および各ストア操作のアドレスの範囲における一致をチェックする比較器から構成されている。

【００３７】図３に示す実施形態では、ある実装が非プログラム順序でロードすることのできるレジスタ数より少ないＡＣエントリを含む場合には、ＡＣエントリのないレジスタは、永久に無効にセットされる関連する有効ビットを有するだけである。このようにして、存在しないＡＣエントリへのアクセスは無効エントリをが報告し、遅延例外がを生成する。

【００３８】非プログラム順序命令は、以下のようにレジスタ・オペランドのＤＸビットを使用する。命令が例外を生成する場合には、目的レジスタのＤＸビットのみがセットされ、例外は発生さ。命令によって使用されたオペランドのＤＸビットがセットされると（これは、遅延例外がすでに生成されたことを示す）、目的レジスタのＤＸビットもセットされる。すなわち、遅延例外が非プログラム順序操作を通して伝播される。

【００３９】上に述べたリソースに加えて、本発明は、プロセッサによって実行される次の命令を含む。Ｌｏａｄ　Ｒｅｇｉｓｔｅｒ　Ｏｕｔ－ｏｆ－ｏｒｄｅｒ　－

13

この命令は、メモリ位置をレジスタにロードし、有効のマークが付けられた対応するＡＣエントリ内のオペランドの実アドレスの範囲をストアする。実際には、ロード命令は、該命令のプログラム順序／非プログラム順序性を示す１ビットを付加することで拡張される。Ｍｏｖｅ Ｒｅｇｉｓｔｅｒ Ｏｕｔ－ｏｆ－ｏｒｄｅｒ －

この命令は、ソース・レジスタの内容とそれに関連するＡＣエントリを目的レジスタ及び関連する目的ＡＣエントリにコピーする。その機能から、これは、常に非プログラム順序命令である。ソース・レジスタの遅延例外（ＤＸ）ビットは、目的レジスタの遅延例外ビットにコピーされるが、例外は発生しない。Ｃｏｍｍｉｔ Ｒｅｇｉｓｔｅｒ － この命令は、プログラム順序でのみ実行される。これは、非プログラム順序でロードされたレジスタの内容を他のレジスタにコピーし、ソース・レジスタに関連するアドレス比較器エントリがまだオーバーラップするアドレスへのストアによって無効化されていないかどうかをチェックし、また、ソース・レジスタの遅延例外ビットがセットされていないかどうかをチェックする。アドレス比較器エントリが有効であり遅延例外（ＤＸ）ビットがセットされていない場合には、レジスター移動操作が進行する。ＡＣエントリが有効でない場合あるいはＤＸビットがセットされている場合には、遅延例外が生成される。Ｉｎｖａｌｉｄａｔｅ Ａｄｄｒｅｓｓ Ｃｏｍｐａｒａｔｏｒ Ｅｎｔｒｉｅｓ －

この命令は、アドレス比較器の全内容を無効化する。これは、コンテキストの切り替え時にＡＣが保管されていない場合に、古いコンテキストからのＡＣが新しいコンテキストにおいて例外を生成するのを避けるためにシステム・ソフトウエアによって使用される。

【００４０】ロード操作の投機の例

以下、上に述べたリソースの使用を例により示す。下の左側の欄は（元の）コードを示すもので、ストア命令の下に、ロード命令と、そのロードに依存する若干の算術命令を含んでいる。この例では、命令名の後の最初のレジスタが目的レジスタであり、残りのレジスタはオペランドである。

元のコード

```
・・・
・・・
ｓｔｏｒｅ　ｒ３，２０（ｒ２）
・・・
ｌｏａｄ　　ｒ５，１０（ｒ４）
ａｄｄ　　ｒ６，ｒ５，２０
ｓｕｂ　　ｒ７，ｒ６，ｒ７
・・・
```

ストアの上に移動されたロード

```
ｌｏａｄ？　　ｒ２５，１０（ｒ４）
・・・
ｓｔｏｒｅ　　ｒ３，２０（ｒ２）
・・・
ｃｏｍｍｉｔ　ｒ５，ｒ２５
ａｄｄ　　ｒ５，ｒ５，２０
ｓｕｂ　　ｒ７，ｒ６，ｒ７
・・・
```

ロード命令は、右側の欄に示すようにストアの上に動かされる。この例では、目的レジスタがリネームされると仮定している。右側の欄のロード命令コードにクエスチョン・マークで示されているように、新しい目的レジスタは非プログラム順序でロードされる。その結果、そのデータは目的レジスタにロードされるが、ロードされたオペランドのアドレスの範囲は、有効のマークが付けられたＡＣエントリに保管される。右側の欄に示すように、元のロード命令は、コミット命令で置換される。

【００４１】ロードに続く算術命令が下に示すように順序替えされると仮定する。

前のコード

```
ｌｏａｄ？　　ｒ２５，１０（ｒ４）
・・・
・・・
・・・
ｓｔｏｒｅ　　ｒ３，２０（ｒ２）
・・・
ｃｏｍｍｉｔ　ｒ５，ｒ２５
ａｄｄ　　ｒ６，ｒ５，２０
ｓｕｂ　　ｒ７，ｒ６，ｒ７
・・・
```

コミット命令の上に移動された操作

```
ｌｏａｄ？　　ｒ２５，１０（ｒ４）
ａｄｄ？　　ｒ２６，ｒ２５，２０
ａｄｄ？　　ｒ２７，ｒ２６，ｒ７
・・・
ｓｔｏｒｅ　　ｒ３，２０（ｒ２）
・・・
ｃｏｍｍｉｔ　ｒ５，ｒ２５
ｃｏｐｙ　　ｒ６，ｒ２６
ｃｏｐｙ　　ｒ７，ｒ２７
・・・
```

さらに、これらの命令が例外を生じないと仮定する。その場合には、コミット操作の後にレジスタ・コピー操作が続き、必要ならば非プログラム順序操作の結果をそれらの宛先レジスタへコピーする。このようなコピー操作は、コンパイラが行なうコードの最適化の中のコピー伝播ステップによってなくすことができる。非プログラム順序レジスターのコミット命令とは異なり、このレジスタ・コピー操作は、対応するオペランドが前のコミット操作によって暗示的に有効化または無効化されているため、アドレス比較器をチェックすることなく、ソース・

15

レジスタを宛先レジスタにコピーするだけである。

【００４２】ストア操作が非プログラム順序でロードされた位置とオーバーラップする場合には、ストア操作の副作用として、対応するＡＣエントリが無効としてマークが付けられる。その結果、コミット命令の実行が遅延例外を生じる。この例外に関連するハンドラは、ロード操作ならびに該ロードに依存しまた例外が生じる前に実

順序替えされたコード

```
load?  r25, 10 (r4)
add?   r26, r25, 20
sub?   r27, r26, r7
...
store  r3, 20 (r2)
...
commit  r5, r25
copy    r6, r26
copy    r7, r27
...
```

非プログラム順序ロードに依存する各命令は、回復コードの一部として再実行され、該回復コードは、コミット操作を再実行するために戻ることに留意が必要である。あるいは、さらに最適化のため、可能な場合には、回復コードは、元の目的レジスタを直接に更新して、コピー伝播の最適化によって除去されなかったレジスタ・コピー操作をスキップすることができる。以上、本発明を好ましい一実施形態に関して説明したが、当業者は、本発明を特許請求の範囲の精神および範囲内で変更して実施できることが理解できよう。

【００４４】まとめとして、本発明の構成に関して以下の事項を開示する。

（１）スーパースカラまたはＶＬＩＷプロセッサにおいてメモリ操作を順序替えする方法であって、プロセッサによって発行された命令をデコードするステップと、デコードされた命令が非プログラム順序ロード命令であるかどうかを判別し、そうであれば、その非プログラム順序ロード命令が例外を生成するかどうかを判別するステップと、例外を生成する非プログラム順序ロード命令について、前記ロード命令の目的レジスタに関連する遅延例外ビットをセットするステップと、例外を生成しない非プログラム順序ロード命令のメモリ・アドレスをアドレス比較器に保管し、前記アドレス比較器に保管されたメモリ・アドレスのための有効ビットをセットするステップと、デコードされた命令がストア操作であるかどうかを判別するステップと、デコードされたストア命令によって参照されたメモリ・アドレスの範囲を前記アドレス比較器の中のすべてのエントリと比較するステップと、前記アドレス比較器の中の一致する各エントリについて、その対応するエントリの有効ビットを無効にセットするステップと、デコードされた命令がコミット操作であるかどうかを判別するステップと、前記デコードされたコミット操作の目的レジスタに関連するアドレス比較器エントリの前記有効ビットをチェックし、前記有効ビットが無効にセットされている場合には遅延例外を生成し、同時に、前記コミット操作のソース・レジスタの遅延例外ビットをチェックし、遅延例外ビットがセットされている場合には、遅延例外を生成するステップと、例外命令を打ち切り、制御を例外ハンドラへ移すステップと、を有する方法。

（２）上記（１）に記載のスーパースカラまたはＶＬＩＷプロセッサにおいてメモリ操作を順序替えする方法であって、さらに、前記デコードされた命令が非プログラム順序ロード、ストア、又はコミット命令以外であるかどうかを判別するステップと、該命令によって用いられたすべてのソース・レジスタのための前記遅延例外ビットをチェックし、任意の遅延例外ビットがセットされている場合には、目的レジスターの対応する遅延例外ビットをセットするステップと、を有する方法。

（３）メモリ操作を順序替えすることのできるスーパースカラまたはＶＬＩＷプロセッサであって、前記プロセッサによって発行される命令をデコードするためのデコーダと、各々が特殊レジスタとしてアクセス可能な遅延例外ビットを有する複数のレジスタと、デコードされた命令が非プログラム順序ロード命令であるかどうかを判別し、そうであれば、該非プログラム順序ロード命令が例外を生成するかどうかを判別する機能的手段であって、例外を生成する非プログラム順序ロード命令について

16

行された二つの操作を実行する回復コードを含んでいる。このため、回復コードの中で再実行される命令のオペランドは、同じレジスタの中または他の位置でまだ利用可能でなければならない。

【００４３】例として、上に示した順序替えコードのための下に示す回復コードを検討する

回復コード

```
rcvr: load   r25, 10 (r
      add    r5, r25, 2
      sub    r5, r26, r
      return
```

17

て前記ロード命令の目的レジスタに関連する遅延例外ビットをセットする機能的手段と、例外を生成しない非プログラム順序ロード命令のメモリ・アドレスを保管するためのアドレス比較器であって、保管されたメモリ・アドレスのためにセットされる有効ビットを有するアドレス比較器と、ただし、前記機能的手段は、デコードされた命令がストア操作であるかどうかを判別し、前記アドレス比較器は、デコードされたストア命令によって参照されたメモリ・アドレスの範囲を前記アドレス比較器の中のすべてのエントリと比較し、前記アドレス比較器の中の一致する各エントリについて、その対応するエントリの有効ビットを無効にセットし、前記機能的手段は、デコードされた命令がコミット操作であるかどうかを判別し、該デコードされたコミット操作のソース・レジスタに関連するアドレス比較器エントリの前記有効ビットをチェックし前記有効ビットが無効にセットされている場合には遅延例外を生成し、また同時に、前記コミット操作のソースレジスタの遅延例外ビットをチェックし、遅延例外ビットがセットされている場合には、遅延例外を生成して例外命令を打ち切り、例外命令が打ち切られるときに回復コードを実行する例外ハンドラーし、を有するスーパースカラまたはＶＬＩＷプロセッサ。

18

【図面の簡単な説明】

【図１】本発明によって行なわれるプログラム順序および非プログラム順序の操作の実行のための論理を示すフローチャート。

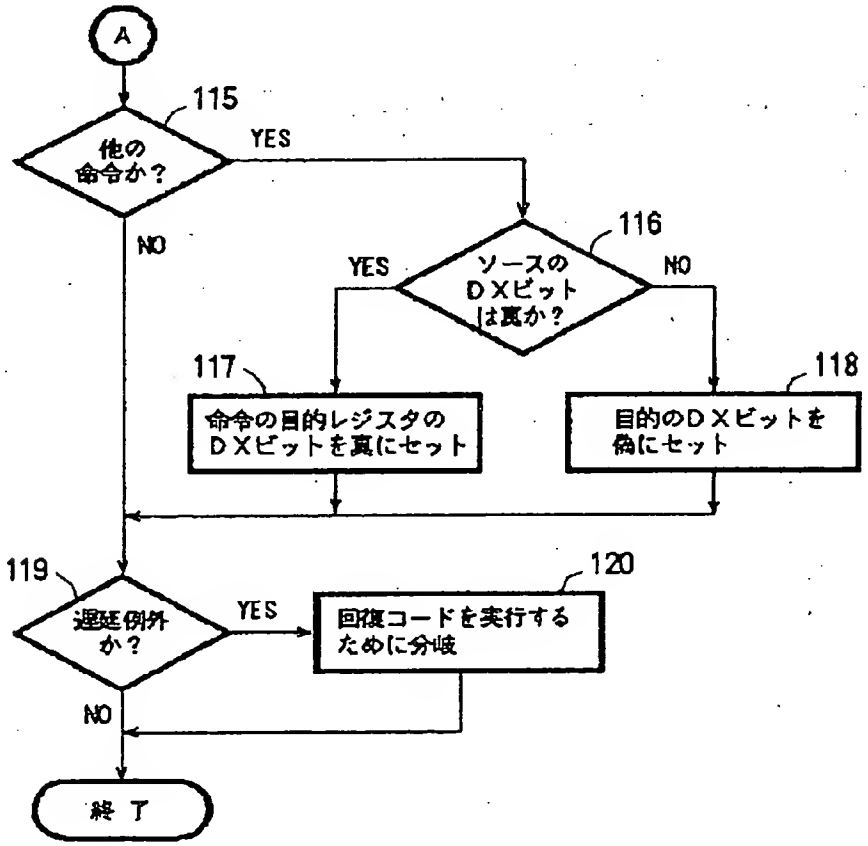
【図２】本発明によって行なわれるプログラム順序および非プログラム順序の操作の実行のための論理を示すフローチャート。

【図３】図１及び図２のフローチャートに示した非プログラム順序の操作の実行をサポートするハードウエア・リソースを示すブロック図である。

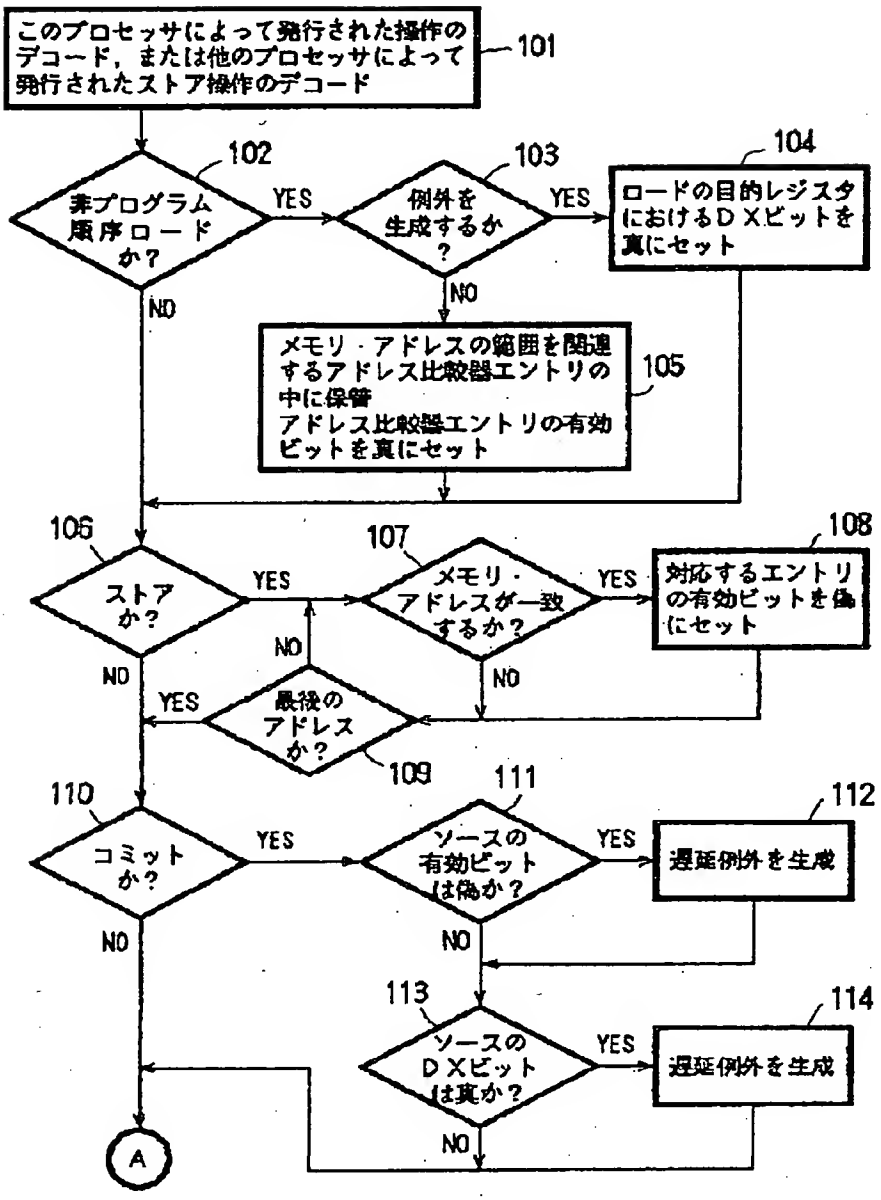
【符号の説明】

２０１～２０６	機能ユニット
２０７	データ・キャッシュ
２０８	汎用レジスタ
２０９	浮動小数点レジスタ
２１０	条件レジスタ
２１１	アドレス比較器
２１２～２１４	ＤＸビット

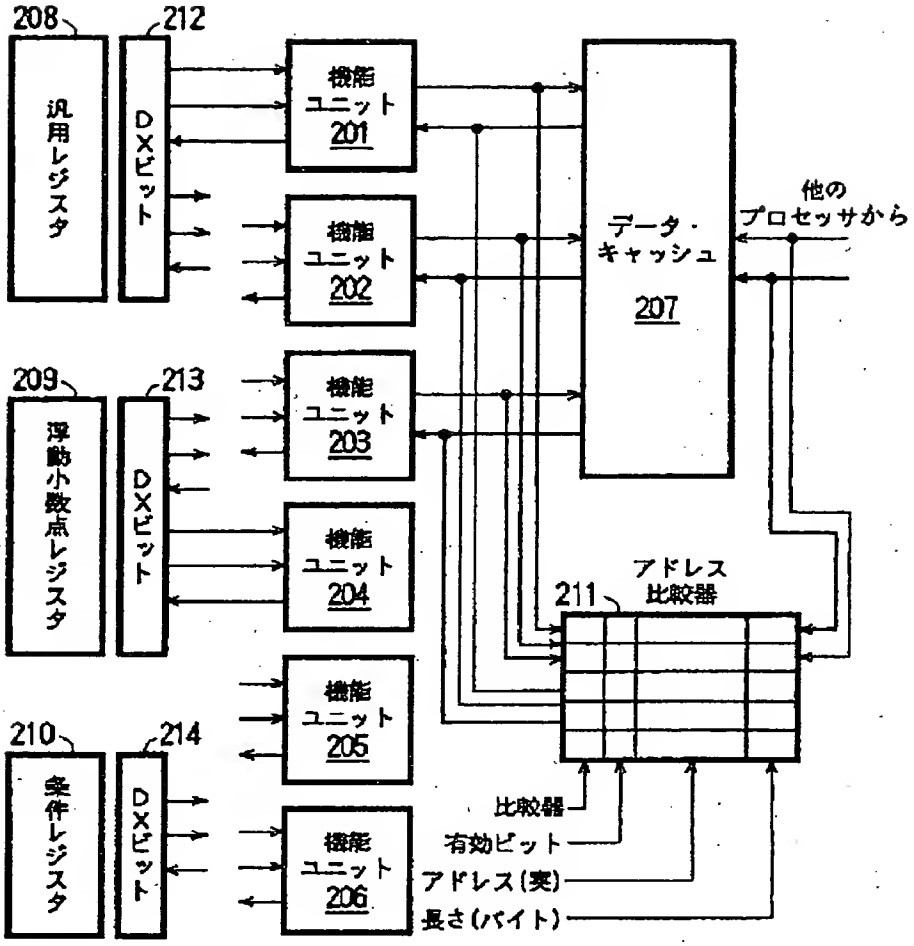
【図２】



【図１】


このプロセッサによって発行された操作のデコード，または他のプロセッサによって発行されたストア操作のデコード
101
非プログラム順序ロードか？
102
YES
例外を生成するか？
103
YES
ロードの目的レジスタにおけるＤＸビットを真にセット
104
NO
メモリ・アドレスの範囲を関連するアドレス比較器エントリの中に保管
アドレス比較器エントリの有効ビットを真にセット
105
ストアか？
106
YES
メモリ・アドレスが一致するか？
107
YES
対応するエントリの有効ビットを偽にセット
108
NO
NO
最後のアドレスか？
YES
109
NO
コミットか？
110
YES
ソースの有効ビットは偽か？
111
YES
遅延例外を生成
112
NO
ソースのＤＸビットは真か？
113
YES
遅延例外を生成
114
NO
A

【図３】



フロントページの続き

(72)発明者　デッビット・アーノルド・ルイック
アメリカ合衆国55906ミネソタ州　ローチェスター　ハウソーン　ヒル　ロード　エヌ・イ　2317

(72)発明者　ジャイミー・ハムベルト・モレノ
アメリカ合衆国10530ニューヨーク州　ハーツデール　ホルメス・アベニュー　205

(72)発明者　ガブリエル・モーリシオ・シルバーマン
アメリカ合衆国10546ニューヨーク州　ミルウッド　ヒッデン　ホーロー・レーン　141

(72)発明者　フイリップ・ブラウン・ウィンターフィールド
アメリカ合衆国55902ミネソタ州　ローチェスター　エイス　アベニュー　エス・ダブリュウ　822